THE BELL SYSTEM TECHNICAL JOURNAL

DEVOTED TO THE SCIENTIFIC AND ENGINEERING ASPECTS OF ELECTRICAL COMMUNICATION

Volume 61 December 1982 Number 10

An Overview of PANACEA, a Software Package for Analyzing Markovian Queueing Networks

By K. G. RAMAKRISHNAN and D. MITRA

(Manuscript received April 19, 1982)

PANACEA is a software package that significantly extends the range of Markovian queueing networks that are computationally tractable. It solves multi-class closed, open, and mixed queueing networks. Based on an underlying theory of integral representations and asymptotic expansions, PANACEA solves queueing networks that are orders of magnitude larger than can be solved by other established algorithms. The package is finding widespread use in Bell Laboratories. It also has important software innovations. A flexible programming-language-like interface facilitates compact representation of large queueing networks. An out-of-core implemental strategy enables PANACEA to be ported to processors with modest memory. The modular structure of this software package, along with the automatic machine-generated parser, makes it easily extendable. This paper provides an overview of two basic versions of PANACEA, versions 1.0 and 1.1, which solve "closed" networks only. A description of its model language is given from the point of view of its capability to describe queueing networks in a compact, natural manner. The paper discusses the algorithms, together with their time and storage requirements, that are used in the implementation. Several numerical examples are given.

I. INTRODUCTION

Closed Markovian queueing networks are acknowledged to be val-

uable tools for analyzing computer systems, computer communication systems, on-line computer networks, and other real-time computer-based systems (Refs. 1 through 12 document many real-life examples). Despite this large collection of problems modeled by closed queueing networks, until recently, only a small number of these models were computationally tractable using established algorithms.[13–17] The tractable models have small numbers of customer classes and a small population in each customer class. PANACEA (Package for Analysis of Networks of Asynchronous Computers with Extended Asymptotics) extends the class of computationally tractable, closed queueing networks by several orders of magnitude.

The solution of "large" queueing networks by PANACEA is made possible by an underlying theory of integral representations and asymptotic expansions of relevant performance measures.[18–20] PANACEA obtains solutions to queueing networks with p customer classes and q processing centers, in $O(qp^t)$ operations and about $20q$ kilobytes of storage. Here "t" denotes the number of terms in the asymptotic expansions of the first and second moments of quantities of interest (utilizations, queue lengths, etc.). Extensive computational experiments demonstrate that four terms ($t = 4$) are usually more than adequate to produce highly accurate results. Most importantly, the number of operations and the amount of storage are both independent of the population sizes of customer classes. In contrast, both the convolutional algorithm and mean value analysis[13–17] require operations and storage proportional to

$$p(1 + 2q) \prod_{j=1}^{p} K_j,$$

where K_j is the population size of customer class j.

Thus, PANACEA is distinguished by its ability to solve exponentially growing problems in polynomial time. There are additional important features to PANACEA:

(*i*) PANACEA incorporates a special-purpose queueing network language designed to describe queueing networks.

(*ii*) PANACEA computes second moments of queue lengths. We know of no other package that provides this information.

(*iii*) All computed measures of network performance are typically accompanied by upper and lower bounds.

(*iv*) A complete complexity count enables PANACEA to guarantee a response time for a problem solution by automatically selecting the maximum number of terms to use in the asymptotic expansions. This, in conjunction with (*iii*) above, is often useful in the early stages of system design.

PANACEA runs interactively on computer systems with a C-lan-

guage compiler and LEX,[21] the automated generator of lexical analyzer. This paper describes two basic versions, 1.0 and 1.1, of the package. Other versions have subsequently been implemented and these will be described elsewhere. Version 1.0 solves closed queueing networks in which customers are not allowed to change classes while circulating in the network (no class hopping). This version is implemented completely in-core. Version 1.1 provides for the "class-hopping" feature of queueing networks. An analysis of class-hopping queueing networks shows very large storage requirements even for networks of moderate size. Hence, in Version 1.1 an out-of-core strategy is employed.

Versions 1.0 and 1.1 each consist of about 6000 lines of portable C code, and currently run on VAX 11/780 hardware. Queueing network problems are solved in a guaranteed response time of at most one minute on a nominally utilized VAX 11/780. Many experiments conducted on PANACEA indicate that it is robust and numerically stable. We should mention that PANACEA is being used currently in several projects within Bell Laboratories. The experiences of the users are guiding the enhancements to PANACEA.

The interface to both versions of PANACEA has been designed to allow the user to describe large queueing networks compactly and in a natural manner. Many of the features of general-purpose programming languages have been incorporated into PANACEA Model Language (referred to as PML, henceforth), thus making it a special-purpose queueing network language. Features of PML, such as the preprocessor variable usage, free use of comments, etc., have been freely borrowed from programming languages.

PANACEA is envisioned to run on hardwares ranging from microprocessors to large mainframes. The expressions derived for time and storage requirements in Section IV enable the user to "package" PANACEA to suit the availability of computing resources such as central processing unit (CPU) cycles and memory. For example, in Motorola 68000 with a 0.2 million instruction per second and 256K physical user memory, the user can solve network problems as large as one in which there are 12 processing centers and 10 job classes, in a guaranteed response time of at most one minute. [See eqs. (15) and (16).]

We mention some of the main limitations of PANACEA in its present form. The queueing networks are required to belong to the class of "product form" networks.[1,4] In addition, "normal usage" must be operative in these networks. Normal usage is defined in Section 3.1. In practice this translates to the requirement that no processing center in the queueing network is utilized more than about 85 percent. Finally, load-dependent service rates are not allowed in the existing versions of PANACEA.

Our discussion of PANACEA is organized into six sections. Section II describes the global properties of PML and the overall software structure of PANACEA. Section III details the internals of the computational phase, together with an overview of the theory of asymptotic expansions. Section IV, "Computational Complexity," counts operations and storage required, and also gives experimental results for validation. Several numerical examples illustrating different features of PANACEA are given in Section V. Section VI presents conclusions and proposed future enhancements to PANACEA.

II. FEATURES OF PML

This section presents an overview of PML. Rigorous syntax and semantic definitions of PML statements have been developed.[22] Properties of PML statements are illustrated by considering a prototypical queueing network shown in Fig. 1. This queueing network represents a service point of a large packet-switched communications network. This service point consists of three front-end processors (terminal concentrators) that act as gateways for the terminals connected to the service point. The front-end processors (FEPs) are connected to three central processing units (CPUs). These processors, when in need of data base access, communicate with a data base processor. The labels within the processors are symbols used in the PML program describing the queueing network. These labels are arbitrary and can be chosen by the user in any manner. The node labeled TERMINAL represents the terminal population connected to the service point. This terminal population is divided into nine customer classes. After obtaining service at any of the CPUs, customer classes 1 through 4 return to the TERMINAL to generate the next command, whereas customer classes 5 to 9 transit to DBP, since they require data base access. Figure 2 gives the PML program that describes the queueing network shown in Fig. 1. The lines in the program are numbered for easy reference to statements. They are not part of the PML program. The following discussion of PML should be read in conjunction with Fig. 2.

2.1 Overview

PML is a free-format language. Line boundaries, tabs, and white spaces in the statements are completely ignored. Users are permitted the flexibility of segmenting statements across multiple lines, indenting portions of their statements, specifying multiple statements on a line, or using a combination of the above. A statement in PML describes some aspect of the queueing network being solved. For example, the statement on line number 6 in Fig. 2 specifies the routing probabilities between TERMINAL and the FEPs. PML also provides the flexibility of symbolic representation of any entity of the queueing network,

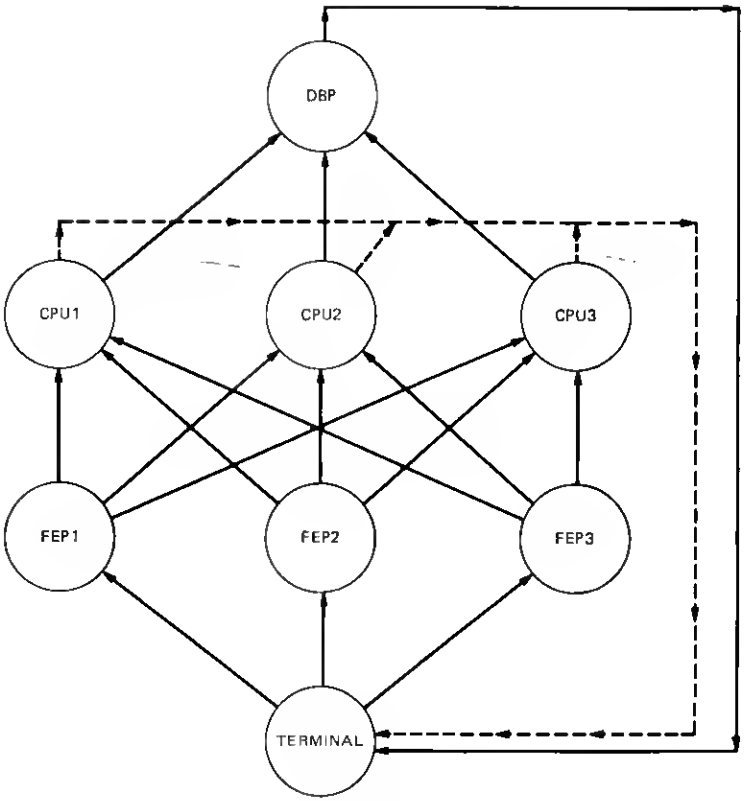


Fig. 1—Model example.

using symbols of arbitrary length. A symbol table manager in the parser module of PANACEA manages these symbols specified by the PML program. Symbols of the PML program become contextually defined as to their data type when they appear in the program. No statements are needed in PML to declare data types for symbols. For example, the symbols "total-classes" and "call-classes" in lines 3 and 4 of Fig. 2 become contextually defined as preprocessor variables of type "integer." Similarly, the symbols "TERMINAL," "FEP1," "FEP2," and "FEP3"—appearing in line 6 of Fig. 2—acquire the data type "string." PML allows arbitrarily complex arithmetic expressions to be specified in the statements. The arithmetic operators in these expressions have the standard precedence and associativity. PML programs can be made readable and self-documenting by making use of the facility for comments embedded in the program. With this general introduction to PML, we will now attempt to describe some of its global properties.

```
 1 /*ILLUSTRATIVE EXAMPLE*/
 2 /*PREPROCESSOR DEFINITIONS*/
 3                          total_classes = 9;
 4                           call_classes = 4;
 5 /*ROUTING DESCRIPTIONS*/
 6 TERMINAL       :FEP1,FEP2,FEP3   1.0/3.0;
 7 FEP1,FEP2,FEP3 :CPU1,CPU2,CPU3   1.0/3.0;
 8 CPU1,CPU2,CPU3:TERMINAL
 9                 {(1 to call_classes) 1.0};
10 CPU1,CPU2,CPU3:DBP
11                 {(call_classes + 1
12                    to
13                  total_classes) 1.0};
14 DBP             :TERMINAL
15                 {(call_classes + 1 to
16                  total_classes) 1.0};
17 /*RATE DESCRIPTIONS*/
18 CPU1,CPU2,CPU3 {12.5,20,12.5,20,2.5,3.3,2.5,3.3,8.33};
19 'FEP.*'        10.0;
20 DBP {(call_classes+1).73611,.73611,.3833,.3823,3.33};
21 TERMINAL {.00333,.00444,.0008333,.01555,
22            .0008333,.0008335,.000076,.000079,.0083};
23 /*DISCIPLINE DEFINITIONS*/
24 CPU1,CPU2,CPU3 PS;TERMINAL IS;DBP PS;
25 FEP1,FEP2,FEP3 FCFS;
26 DEGREE {(1 to 4) 75,150,150,5,5,180};
27 OUTPUT DBP UTIL + QLENGTH + Q2LENGTH;
28 END;
```

Fig. 2—PML program.

2.2 Global properties

PML is characterized by the global properties of compactness, extendability and self-documentation, as discussed below.[23]

2.2.1 Compactness

A programming language is considered compact if the user can specify "aggregate" entities using a single "atomic" entity descriptor. PML enables the user to specify large networks compactly. Many features of the language aid in this compactness.

2.2.1.1 Multiple edge and node descriptors. All entities in the network that have similar characteristics are described together in a single statement. For example, the statement in line 7 of Fig. 2 states that the transition probability for *all edges* emanating from the front-end processors and terminating in the node processors is 1/3. Thus, 81 edges are described in a single statement, as explained below.

The node symbols separated by commas and delimited by ":" denote the set of originating nodes. The comma-separated node symbols specified after the ":" denote the set of destination nodes. Thus, in

Fig. 2, every origin-destination pair specifies nine edges (one for each customer class). As a universal rule in PML, aggregate names (node names separated by commas) can be given wherever a single node name is expected. The operand specification is associated with every node in the aggregate name. For instance, the service rate descriptor in line 18 of Fig. 2 specifies all the service rates of the CPUs in a single statement.

2.2.1.2 Implicit class propagation. When many customer classes have identical characteristics, the user specifies the characteristics exactly once. All customer classes implicitly obtain those characteristics, thus resulting in compactness. If we refer back to line 7 in Fig. 2 again, we can see that the transition probability 1/3 specified in the operand propagates to all nine customer classes. If the customer classes differ in their characteristics, then a separate value can be specified for each class, as line 18 shows. The point to note here is the implicit class indexing agreed upon between the parser and the PML programmer. Another feature of PML allows the user to group some classes that have identical characteristics by prefixing the operand value by a group range, as we see in lines 9, 15, and 16. In line 9, the expression enclosed in angle brackets "⟨" and "⟩" specifies the class range to be 1 to 4 (recall that call-classes = 4), and a value of 1.0 is assigned to the probability of transition for these classes. The class indices outside the range are unaffected. Thus, the user has the flexibility of enumerating each class, grouping classes, or omitting the specification for the classes altogether.

2.2.1.3 MACROS, INCLUDE FILES and regular expressions. PML provides many language tools to condense frequently occurring sequences of statements. MACROS and INCLUDE FILES enable PML programs to be compact. PML MACROS and INCLUDE FILES follow standard conventions of MACRO and INCLUDE FILE specification of programming languages. Another tool that promotes compactness is regular expressions. Any collection of symbols having common subsymbols can be specified by a single regular expression. For instance, the regular expression 'FEP.*' in line 19 of Fig. 2 is a regular expression that specifies the three nodes FEP1, FEP2, and FEP3.

2.2.2 Extendability

The PML parser and lexical analyzer have been machine generated. Once PML grammar rules and lexical tokens are specified, the parser and the lexical analyzer can be generated by existing automated tools (Yacc[24] and Lex[21]). This implies that it is trivial to extend or modify the language by simply respecifying the grammar rules and/or the lexical tokens. PANACEA is designed to be an integrated queueing network package with the ability to handle most classes of queueing

networks. The extendability feature of PML is crucial in enhancing the descriptive nature of PML, to describe, for instance, open networks, priority networks, etc.

2.2.3 Self-documentation

PML aids in self-documenting the model, by the use of comments anywhere in the model, preprocessor variables, and indentation and structuring of the model. We have deliberately omitted details of each statement in the PML program of Fig. 2, in the hope that the self-documenting feature of PML will obviate this necessity.

2.3 Structure of PANACEA

Figure 3 depicts the structure of PANACEA software. The input phase of PANACEA consists of the lexical analyzer and the parser. These two modules work in lockstep, parsing statements of the user program (written in PML), validating the statements, and gleaning the essential information from these statements. The object module that is created is then consumed by the control module, which is table driven. The asymptotic series for each quantity of interest (utilization, mean queue length, second moments on queue length, etc.) is encoded in a table. The table contains information on how to compute the elements of the series in terms of the moment partition functions of the pseudo-networks (see Section III). These pseudo-networks are solved by the pseudo-network modules using recursive techniques. Finally, all relevant quantities of interest requested by the user are displayed as output by the output module. PANACEA has been modularly structured in this fashion, so that advanced users can bypass one or more of the three phases of PANACEA (see Fig. 3), thus making the execution faster. For example, an object module that has been previously created can be directly passed on to the computation phase, bypassing the compilation phase.

The above discussion is relevant to both Versions 1.0 and 1.1. However, Version 1.1, which implements the class-hopping feature of queueing networks, has an additional feature. The storage of the routing matrices and the process of solving for their Perron eigenvector are done out-of-core. A "MAP" is kept in-core denoting the exact position, in secondary storage, of the matrix element $P_{(\sigma,\tau)(j,i)}$ (transition probability of going from node τ to node i while hopping from customer class σ to customer class j). Whenever this element needs to be retrieved or stored, PANACEA performs an input/output operation to the appropriate place in secondary storage. Naturally, a compromise is made in sacrificing run time while gaining significant reduction in virtual memory. The run time can be improved by "optimal" caching of matrix elements, according to the availability of physical memory.

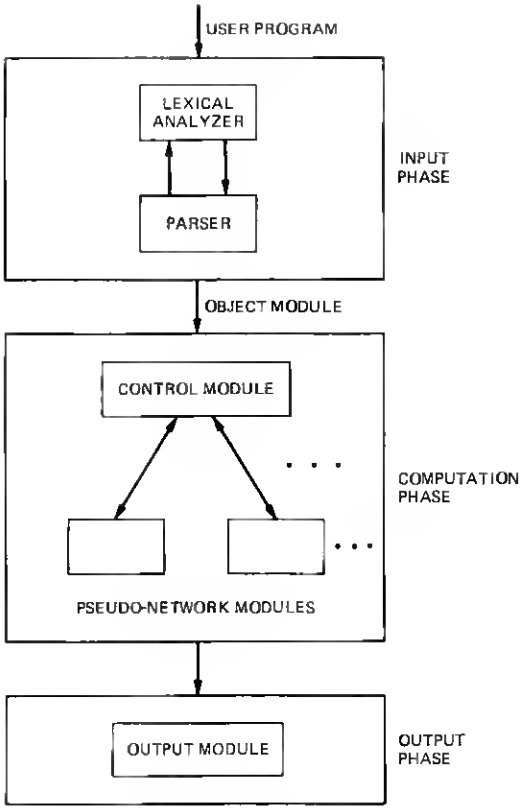


Fig. 3—Structure of PANACEA.

III. THE COMPUTATIONS

3.1 Background to asymptotic expansions

Our description of the background to asymptotic expansions shall be brief. We begin with the main results arrived at in the companion papers.[18–20] In addition, we restrict our descriptions to the first and autocorrelative second moments of individual queue lengths in the processors; various other quantities, such as throughput and processor utilization, are also computed in PANACEA but omitted in the discussion as they are either simply derived from or closely related to the quantities discussed.

In the network

$$p = \text{number of classes of jobs,} \tag{1}$$

$$q = \text{number of active (i.e., of types 1, 2, and 4) processing centers}^4 \tag{2}$$

and let the classes and centers be indexed by σ and τ, respectively. Let $n_{\sigma\tau}$ denote the random number of jobs of class σ in center τ. For the leading moments from the stationary distributions of individual processor queue lengths, we have from Refs. (19) and (20) the following *exact* expressions:

$$\left.\begin{aligned} \langle n_{\sigma\tau}\rangle(\mathbf{K}) &= \frac{r_{\sigma\tau}K_\sigma}{\alpha_\tau}\frac{I^{(1)}_{\sigma,\tau}(N)}{I(N)} \\ \langle n_{\sigma\tau}(n_{\sigma\tau}-1)\rangle(\mathbf{K}) &= \frac{r^2_{\sigma\tau}K_\sigma(K_\sigma-1)}{\alpha^2_\tau}\frac{I^{(2)}_{\sigma,\tau}(N)}{I(N)} \end{aligned}\right\} \begin{matrix} 1\le\sigma\le p, \\ 1\le\tau\le q \end{matrix} \tag{3}$$

where

K_σ = population of jobs of class σ

$r_{\sigma\tau} = \rho_{\sigma\tau}/\rho_{\sigma 0}$

$$\rho_{\sigma\tau} = \frac{\text{relative number of visits of class } \sigma \text{ jobs to center } \tau}{\text{service rate of class } \sigma \text{ jobs in center } \tau}$$

$$\rho_{\sigma 0} = \sum\left[\frac{\text{relative number of visits of class } \sigma \text{ jobs to center } \tau}{\text{service rate of class } \sigma \text{ jobs in center } \tau}\right],$$

where the sum is over all infinite server centers τ visited by class σ jobs;

$$\alpha_\tau = 1 - \sum_\sigma K_\sigma r_{\sigma\tau} \ (>0 \text{ in "normal usage"})$$

$I^{(m)}_{\sigma,\tau}(N)$ = integrals parameterized by N

$$m = 0, 1, 2; \quad 1\le\sigma\le p; \quad 1\le\tau\le q$$

$I(N) = I^{(0)}_{\sigma,\tau}(N)$ (for $m = 0$ there is no dependence on σ or τ)

N = large parameter.

PANACEA chooses

$$N = 1/\min_{\sigma,\tau} r_{\sigma\tau}. \tag{4}$$

While this choice is not at all critical, theoretical reasons corroborated by computational experience indicate that it serves very well to keep well within machine range all terms calculated in the expansions to be described below.

Thus, the quantities requiring computation are the integrals and the theory has developed the asymptotic expansions

$$I^{(m)}_{\sigma,\tau}(N) \sim \sum_{k=0}^{t-1} A^{(m)}_{\sigma,\tau,k}/N^k. \tag{5}$$

Let us digress on

$$t = \text{number of terms in asymptotic expansion.} \tag{6}$$

The error in the calculation of the integrals, and therefore of the queue length moments in (3), from using only t terms is $0(1/N^t)$. In PANACEA, generally,

$$t \leq 4. \tag{7}$$

[For $I(N)$ only, the maximum number of terms is 5.] However, PANACEA has the facility to automatically select t to be less than the maximum. There are two reasons for this. First, if the user desires a guaranteed response time, our detailed complexity count (see Section IV) allows PANACEA to satisfy this by calculating the appropriate value of t. As all output quantities are typically accompanied by lower and upper bounds (see Section 3.6), this facility leading to small t is both useful and often exercised. A second reason is that PANACEA looks for departure from monotonicity of the series in (5), and in the event of its occurrence it truncates the series optimally. This a rare phenomenon and always occurs, as the theory explains, for small networks when N given in (4) is not large enough.

The following resumes the discussion on the computation of the expansion coefficients $A^{(m)}_{\sigma,\tau,k}$.

3.2 *The pseudo-networks in the computation of the expansion coefficients*

The expansion coefficients $\{A^{(m)}_{\sigma,\tau,k}; 0 \leq m \leq 2, 1 \leq \sigma \leq p, 1 \leq \tau \leq q, 0 \leq k \leq t-1\}$ have been completely specified in Refs. 19 and 20 as simple algebraic combinations of other basic quantities $g^{(m)}_{\tau}(\mathbf{n})$. For example,

$$\begin{aligned} A^{(1)}_{\sigma,\tau,2} = {} & 2\sum_j \beta_j g^{(1)}_{\tau}(3\mathbf{e}_j) + \frac{1}{8}\sum_j \beta_j^2 g^{(1)}_{\tau}(4\mathbf{e}_j) \\ & + \frac{1}{2}\sum_{j,k} \beta_j \beta_k g^{(1)}_{\tau}(2\mathbf{e}_j + 2\mathbf{e}_k) + 3\beta_\sigma g^{(1)}_{\tau}(3\mathbf{e}_\sigma) \\ & + \sum_{j \neq \sigma} \beta_j g^{(1)}_{\tau}(\mathbf{e}_\sigma + 2\mathbf{e}_j) + 2g^{(1)}_{\tau}(2\mathbf{e}_\sigma), \end{aligned} \tag{8}$$

in which j and k are distinct class indices, $\mathbf{e}_j$ is the vector with 1 in the jth location and 0 elsewhere, and $\beta_j = K_j/N$. (For $m = 0$, $A^{(m)}_{\sigma,\tau,k}$ is independent of σ, τ and also denoted by A_k.)

It turns out that the underlying quantities $g^{(m)}_{\tau}(\mathbf{n})$ are the mth moment partition functions[20] of a hypothetical network with population vector $\mathbf{n}$. This latter network is related to the original network in having the same number, q, of processing centers but has no infinite

server center and has quite different processing rates. The two important features to note of the population vectors $\mathbf{n}$ that appear as arguments of $g_\tau^{(m)}$ in formulas such as (8) are that

$$\text{at most } t \text{ classes have nonzero population,} \tag{9a}$$

$$\text{the total population, } \sum n_o \leq 8. \tag{9b}$$

For the worst case $t = 4$, we represent

$$\mathbf{n} = n_i\mathbf{e}_i + n_j\mathbf{e}_j + n_k\mathbf{e}_k + n_\ell\mathbf{e}_\ell, \qquad n_i + n_j + n_k + n_\ell \leq 8, \tag{10}$$

where (i, j, k, ℓ) are class indices of the original network and thus $1 \leq i, j, k, \ell \leq p$. Without loss of generality we consider only $1 \leq i < j < k < \ell \leq p$.

For a particular choice of a 4-tuple (i, j, k, ℓ) we therefore have a hypothetical network with only four classes. We call such a restricted hypothetical network a pseudo-network. Clearly, there are as many as $\binom{p}{4}$ pseudo-networks in the worst case and $\binom{p}{t}$ in general. However, each pseudo-network is small.

To summarize, PANACEA computes t ($t \leq 4$) terms in the expansions of each of $(1 + 2pq)$ integrals, and the expansion coefficients $A_{\sigma,\tau,k}^{(m)}$ are computed from the moment partition functions of the pseudo-networks of which there are $\binom{p}{t}$, each with t classes and a total population over all classes of at most 8.

3.3 The pseudo-network computations

PANACEA solves for the mth moment partition functions $\{g_\tau^{(m)}(\mathbf{n})\}$ for each of the pseudo-networks in turn where each pseudo-network is characterized by a leading t-tuple from (i, j, k, ℓ), $1 \leq i < j < k < \ell \leq p$. The 0th moment partition function is independent of τ, i.e.,

$$g_\tau^{(0)}(\mathbf{n}) = g(\mathbf{n}), \tag{11}$$

which is the usual partition function in the literature. The computation of $g(\mathbf{n})$ is done by the established convolutional algorithm.[17] For moments $m = 1, 2$ the computation is done by specializing the recursions given in Ref. 20 to pseudo-networks.

It is noteworthy that for PANACEA we have devised a scheme for implementing the recursion in which only those $g_\tau^{(m)}(\mathbf{n})$ for which $\sum n_o \leq 8$ are computed.

As each pseudo-network has t job classes there are $\binom{8+t}{t}$ such vectors $\mathbf{n}$, and $(1 + 2q)\binom{8+t}{t}$ computed values of $\{g_\tau^{(m)}(\mathbf{n})\}$.

In the computational phase of PANACEA the set of pseudo-net-

works is analyzed only once to compute all the expansion coeffcients $A^{(m)}_{\sigma,\tau,k}$. This implementation relies on $t(1 + 2pq)$ registers to be set up at the outset of the computational phase, one for each of the expansion coefficients. On completion of the computations pertaining to any pseudo-network, the computed data is used to update the relevant registers.

The implementation described above attempts to minimize not only arithmetical operations, but also virtual storage requirements.

3.4 Error bounds

A basic feature of PANACEA is that all computed quantities are accompanied by upper and lower bounds. This is made possible by the following theoretical result[19,20]:

$$I^{(m)}_{\sigma,\tau}(N) \geq \sum_{k=0}^{t-1} A^{(m)}_{\sigma,\tau,k}/N^k \quad \text{if } t \text{ is even},$$
$$\leq \sum_{k=0}^{t-1} A^{(m)}_{\sigma,\tau,k}/N^k \quad \text{if } t \text{ is odd}. \tag{12}$$

PANACEA computes the upper (lower) bound for the moments of the individual queue lengths given in (3) by using odd and even (even and odd) numbers of terms, respectively, in the asymptotic expansions for the integrals in the numerators and denominators of the expressions in (1). In certain cases (see Section 3.1) only one of the usual pair of bounds can be computed and in other, even rarer, cases no bounds can be computed.

IV. COMPUTATIONAL COMPLEXITY

We give a count of the total number of multiplications and the storage requirements for a problem solution on PANACEA. As multiplications are overwhelmingly more time-consuming than additions in double-precision floating-point operations, we omit a count of the latter.

4.1 Multiplications

Consider first the multiplications in the analysis of a particular pseudo-network. We have already noted in Section 3.4 that the number of vectors $\mathbf{n}$ that are valid arguments of $g^{(m)}_{\tau}(\cdot)$ is $\binom{8+t}{t}$. For each vector $\mathbf{n}$ as argument the convolutional algorithm for computing $g(\mathbf{n})$ requires tq multiplications, and to calculate $g^{(m)}_{\tau}(\mathbf{n})$, $m = 1, 2$ and all τ, there are $2tq$ additional multiplications. Thus,

$$\text{multiplications per pseudo-network} = (tq + 2tq)\binom{8+t}{t}. \tag{13}$$

Since only $\binom{p}{t}$ pseudo-networks are analyzed,

$$\text{multiplications total} = \binom{p}{t} 3tq \binom{8+t}{t} \tag{14}$$

$$\sim O(qp^t) \tag{15}$$

as q and p become large.

The remaining major component of the computations, the updatings of the registers (see Section 3.5), requires a total of less than $280\binom{p}{t}$ multiplications, which is usually negligibly small compared with the number in (14).

4.2 *Storage*

Again, consider first the requirement for a particular pseudo-network. The actual storage used in PANACEA is somewhat more than the number of computed quantities and is

$$(1 + 2q)10^4. \tag{16}$$

No further storage is required for the computations of all the pseudo-networks. There are also the $t(1 + 2pq)$ registers, a small number relative to (16).

4.3 *Experimental results*

We have conducted experiments on PANACEA to verify these results. Figure 4 presents PANACEA's response time for a particular network (see Fig. 4a) in which the number of terms in the asymptotic expansions $t = 4$. The results are for various values of the population **K** (Fig. 4b), the number of classes (Fig. 4c), and the number of active processing centers q (Fig. 4d). The broad features of the results on the response times are in agreement with the above complexity analysis: most importantly, **K** does not affect the response time and the dependencies on p and q are like p^4 and q.

V. NUMERICAL EXAMPLES

Several computational experiments have been conducted on PANACEA. These experiments were designed to highlight specific features of the package, explore its limitations, and to identify numerical stability issues. To date, we have not encountered difficulties with overflow or underflow in computing the asymptotic series. Hence, the rescaling of problem parameters that is often required to avoid numerical instability in the established algorithms is not at all an issue

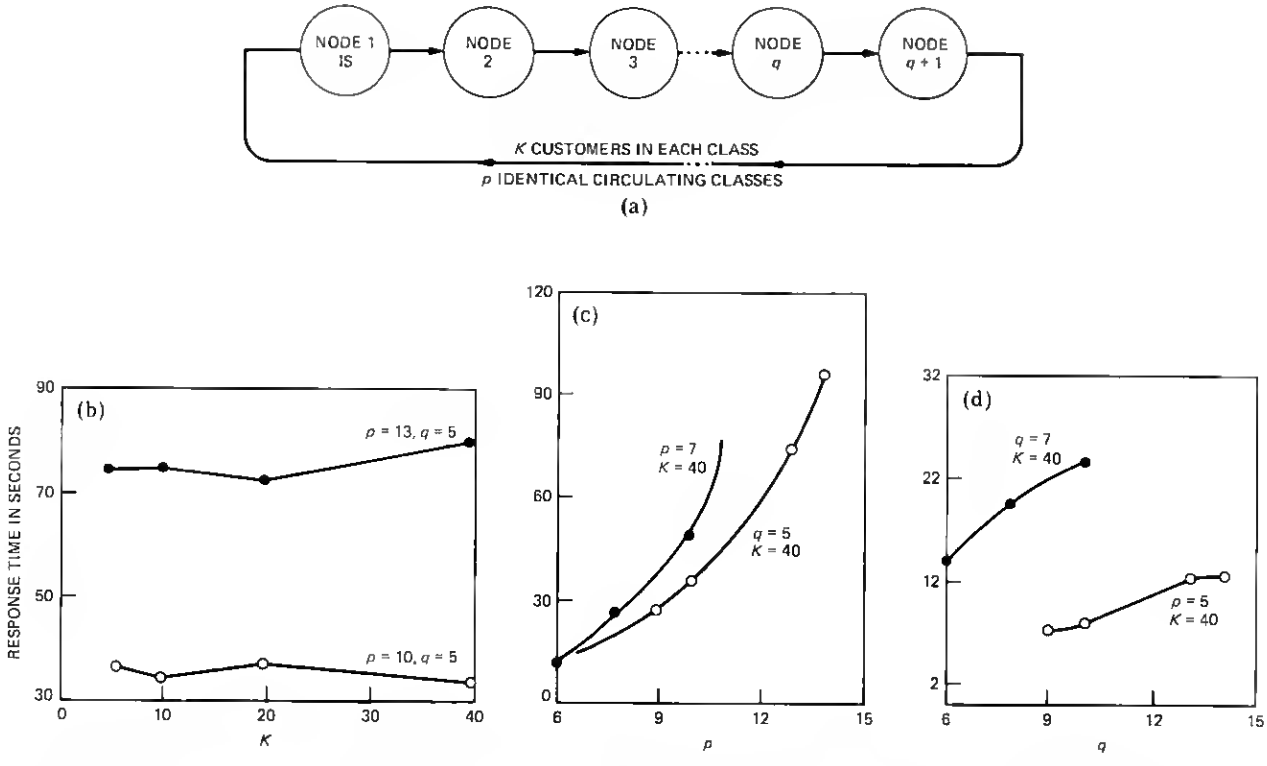


Fig. 4—(a) The network on which the experiments were conducted. Time complexity results for: (b) various values of the population **K**; (c) the number of classes; (d) the number of active processing centers, q.

here. The package has so far performed in a very stable and robust fashion in solving queueing network problems of widely differing sizes.

In all our computational experiments, the queueing network problems ranged in size from very small (one customer class with ten members in the class) to very large (17 customer classes with 17,000 total members in the customer classes). All problems were real-life problems arising in modeling large communication networks. The nature of the interaction of the performance analyst with the package involved solving a given problem, making quick parameter changes in the model, resolving the problem, and so on. This iterative mode of interaction implies the ability of the package to give fast response times, irrespective of the size of the problem. PANACEA was able to ensure a response time of less than 70 seconds for all problems by adaptively truncating the asymptotic series. (All problems were solved on VAX 11/780 with the *UNIX** operating system, Version 3.0, and lightly loaded <60%.) For reasons of brevity, we elaborate only on five numerical examples.

5.1 Example 1

The seven-node, nine-class queueing network shown in Fig. 1 (see the PML program shown in Fig. 2 for degrees of multiprogramming, service rates, etc.) was solved by PANACEA with a response time of about 20 seconds. Figure 5 shows the output displayed by PANACEA corresponding to the node DBP. Several features of Fig. 5 are noteworthy. First, PANACEA prints the number of terms computed in each asymptotic series. In solving this problem, three terms were computed. Second, the bounds printed for all quantities of interest are very "tight." For example, the DBP utilization for customer class 9 is 44.412201 and the upper bound on this utilization is 44.4305312 (an error of at most 0.05 percent). The standard deviations on queue lengths of all customer classes in the node DBP again show the "tight" nature of the bounds. Hence, with three terms in the series, a near-exact solution was obtained for the problem. This implies that for this particular problem, with a given set of parameters, the series converged rapidly.

5.2 Example 2

For some queueing network problems, one may not be able to achieve rapid convergence. This typically happens for "small" networks. Truncating the series too early may result in unacceptable errors. To demonstrate this phenomenon of slow convergence, we modeled the queueing network shown in Fig. 1 with one customer class

* Trademark of Bell Laboratories.

COMPUTATION PHASE BEGINS.
PACKAGE USES 3 TERMS IN THE ASYMPTOTICS FOR GOOD RESPONSE TIME ON THE VAX 11/780 HARDWARE
UTILIZATION STATISTICS ON PROCESSING NODES

UTILIZATION STATISTICS FOR NODE DBP

CLASS	UTILIZATION	UPPER BOUND	LOWER BOUND
1	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
2	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
3	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
4	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
5	1.68863533e − 01	1.68890352e − 01	1.68863533e − 01
6	1.68879473e − 01	1.68906161e − 01	1.68879473e − 01
7	9.95496911e − 04	9.95522116e − 04	9.95496911e − 04
8	1.03167269e − 03	1.03169971e − 03	1.03167269e − 03
9	4.44122010e − 01	4.44305312e − 01	4.44122010e − 01

MEAN QUEUE LENGTH STATISTICS

STATISTICS FOR NODE DBP

CLASS	MEAN QUEUE LENGTH	UPPER BOUND	LOWER BOUND
1	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
2	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
3	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
4	0.00000000e + 00	0.00000000e + 00	0.00000000e + 00
5	7.61035551e − 01	7.61035551e − 01	7.37862183e − 01
6	7.61104587e − 01	7.61104587e − 01	7.37942427e − 01
7	4.50140929e − 03	4.50140929e − 03	4.37652979e − 03
8	4.66486873e − 03	4.66486873e − 03	4.53537014e − 03
9	1.99251497e + 00	1.99251497e + 00	1.92234003e + 00

STANDARD DEVIATION STATISTICS ON QUEUE LENGTH

STATISTICS FOR NODE DBP

CLASS	STD. DEVIATION	UPPER BOUND	LOWER BOUND
1	0.00000000e + 00	—	—
2	0.00000000e + 00	—	—
3	0.00000000e + 00	—	—
4	0.00000000e + 00	—	—
5	1.14408350e + 00	1.15916418e + 00	1.08306158e + 00
6	1.14414970e + 00	1.15922376e + 00	1.08317002e + 00
7	6.71791109e − 02	6.71873612e − 02	6.62227374e − 02
8	6.83911390e − 02	6.83998675e − 02	6.74162160e − 02
9	2.39306393e + 00	2.44979369e + 00	2.16185701e + 00

Fig. 5—Output of model example.

(this class required the services of DBP) with 20 terminals in the class. Table I shows the variation in the utilization of DBP as the number of terms in the asymptotic series is varied. Assuming that after accumulation of five terms, the series has converged to the exact solution, we see that truncating the series after the first term results in a 7-percent error, truncating the series after the second term results in a −4.4-percent error, and so on. Notice the oscillatory nature of convergence. After accumulation of four terms, we still have an error in the units position. Typically, this slow convergence occurs for "small" problems, and hence one can potentially compute more terms in the series without adversely increasing the time complexity. It is interesting to

Table I—Benefit of multiple terms

Number of Terms Used in Expansion	DBP Utilization	Error (%)
1	55.56	+7.0
2	49.63	−4.4
3	53.91	+3.8
4	50.68	−2.4
5	51.92	0

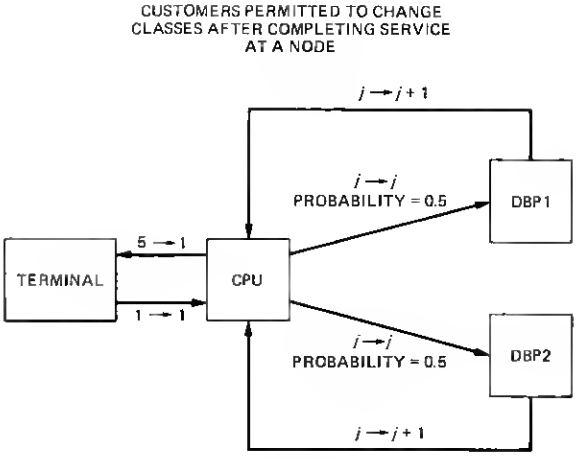
observe that slow convergence is an inherent property of the problem that cannot be cured by large choice of N. At first sight it might appear that simply by choosing N large enough, the convergence of the series may be speeded up. A closer examination of the terms of the asymptotic series reveals the fact that these terms are independent of N and depend only on the initial problem parameters.

5.3 Example 3

Table II shows results of comparing our package with a commercially available package, CADS.[25] CADS is marketed by Information Research Associates and it uses the convolutional algorithm to solve the queueing network. Again, the prototypical network shown in Fig. 1 was solved with one customer class and ten terminals in that class. Attempts at increasing the degree of multiprogramming or increasing the number of classes in the problem resulted in a breakdown of CADS. Hence, extensive comparisons of PANACEA and CADS on large problems could not be accomplished. Table II shows the results for a particular problem solved by both CADS and PANACEA and substantial agreement is exhibited. We have also validated PANACEA and CADS on many other "small" networks. On "small" networks with more than one class and about 10 terminals in all the classes, a significant improvement in the response time of PANACEA was observed, while near-identical results were printed by both PANACEA and CADS. This, of course, is corroborated by our time-complexity analysis (Section IV).

Table II—Comparison with CADS

	CADS		PANACEA	
Processor	Utilization (%)	Queue Length	Utilization (%)	Queue Length
DBP	26.1	0.338	26.08	0.3365
CPU1	4.35	0.0452	4.34	0.0452
FEP1	4.35	0.0452	4.34	0.0452



PROGRAM BEHAVIOR

ATOMIC OPERATION	CPU TIME IN MILLISECONDS	DBP TIME IN MILLISECONDS
OPEN FILE SYSTEM	200	80
OPEN FILE	50	40
READ	10	40
CLOSE	10	40

Fig. 6—Class-hopping model.

5.4 Example 4

This example illustrates the inter-class-transition feature of PANACEA. Version 1.1 of PANACEA permits "class hopping" between customer classes. Models with class-bopping features are the natural tools for analyzing precedence-constrained sequences of actions of a customer in a computer system. Figure 6 shows the central server model in which a terminal-generated request goes through a series of transactions in a strict sequence. Initially, the terminal request is processed by the CPU by executing a program. The germane aspects of the program behavior are captured in four atomic data base operations. Each data base operation set-up time in the CPU and the access time in either of the data base processors, denoted by DBP1 and DBP2, are shown in Fig. 6. When the terminal request is processed by the program, a series of data base transactions is generated to either of the DBPs with equal probability. The transactions, by their inherent nature, have to be executed in the following strict sequence: open file system, open file, read, and close. This precedence is enforced in the model by class hopping. The open-file-system transaction and the terminal request are both assigned a customer class label of 1. The other three atomic data base transactions are assigned class indices 2, 3, and 4, respectively. Class 5 is a "clean-up" class to gracefully

```
1 /*CLASS HOPPING EXAMPLE*/
2 OUTPUT CPU util + q1ength + q2length;
3 /*ROUTING*/
4 CLASS TRANSITION          {1,2,3,4,5};
5 TER          :CPU         {(1)1.0};
6 CPU          :TER         {(5){(1)1.0}};
  CPU          :DBP1,DBP2   {(1 to 4).5};
  DBP1,DBP2    :CPU
                {{(2)1.0},{(3)1.0},{(4)1.0},{(5)1.0}};
  /*SERVICE RATES*/
  CPU          {1000/200,1000/50,1000/10,1000/10,1000/5};
  DBP1,DBP2    {1000/80,(2 to 4)1000/40};
  TER          {(1).05};
  CPU PS; TER IS;DBP1,DBP2 PS;
  DEGREE       {(1)40};
  END;
```

Fig. 7—PML program describing the class-hopping model.

terminate the terminal request. Now, all customers retain their class identity in all but three transitions. When a transition is made from DBP1 or DBP2 to CPU, a hop occurs from class j to class $j + 1$, as shown in Fig. 6 ($1 \leq j \leq 4$). Customer class 5 hops back to class 1 when transiting from the CPU to the terminal.

Figure 7 shows the PML program describing this model. A statement in line number 4 indicates class indices among which hops can take place. The routing statements appropriately denote the "from" class and the "to" class, if necessary.

Figure 8 shows a snapshot of the output produced by PANACEA. The upper and lower bounds are observed to be "tight" for all the quantities displayed in Fig. 8. The response time in solving this problem by PANACEA was almost instantaneous, once the out-of-core phase was completed. The out-of-core phase itself consumed about 60 seconds of elapsed time, on a lightly loaded VAX 11/780.

5.5 Example 5

We now consider the large communication network shown in Fig. 9, which consists of 23 processors, 17 classes and 1,000 terminals in each class. For reasons of brevity, we omit further description of this network. However, one can easily appreciate the explosion in state space of this queueing network and the CPU time that would be required to solve this problem by tbe previous methods. PANACEA was able to solve this problem almost instantaneously (response time was not perceived by the user). Figure 10 shows the output produced by PANACEA. Notice that only one term was computed for each asymptotic series. This implies, from the error analysis, that one of the bounds could not be computed for mean queue length and standard

COMPUTATION PHASE BEGINS.

PACKAGE USES 4 TERMS IN THE ASYMPTOTICS FOR GOOD RESPONSE TIME ON THE VAX 11/780 HARDWARE

UTILIZATION STATISTICS ON PROCESSING NODES

UTILIZATION STATISTICS FOR NODE CPU

CLASS	NODE UTILIZATION	UPPER BOUND	LOWER BOUND
1	3.84754302e−01	3.84754302e−01	3.84336505e−01
2	9.61885755e−02	9.61885755e−02	9.60841263e−02
3	1.92377144e−02	1.92377144e−02	1.92168245e−02
4	1.92377144e−02	1.92377144e−02	1.92168245e−02
5	9.61885719e−03	9.61885719e−03	9.60841227e−03

MEAN QUEUE LENGTH STATISTICS

STATISTICS FOR NODE CPU

CLASS	MEAN QUEUE LENGTH	UPPER BOUND	LOWER BOUND
1	7.65705297e−01	7.93131456e−01	7.65705297e−01
2	1.91426324e−01	1.98282864e−01	1.91426324e−01
3	3.82852634e−02	3.96565713e−02	3.82852634e−02
4	3.82852634e−02	3.96565713e−02	3.82852634e−02
5	1.91426317e−02	1.98282857e−02	1.91426317e−02

STANDARD DEVIATION STATISTICS ON QUEUE LENGTH

STATISTICS FOR NODE CPU

CLASS	STD. DEVIATION	UPPER BOUND	LOWER BOUND
1	1.08703258e+00	1.19926103e+00	1.06718645e+00
2	4.66285569e−01	4.88468353e−01	4.63411440e−01
3	1.98305571e−01	2.03148498e−01	1.98035895e−01
4	1.98305571e−01	2.03148498e−01	1.98035895e−01
5	1.39293183e−01	1.42237373e−01	1.39197236e−01

END OF OUTPUT STATISTICS.

process terminated
*O
Alice:

Fig. 8—Output of Example 4.

deviation statistics. Nevertheless, bounds on the utilization could be computed and they show acceptable accuracy. For example, the maximum percentage error occurs for customer class 14 in the node "hsc2" and it is less than 3 percent. Higher accuracy can be obtained only by computing the second element of the series at the expense of significant increase in response time.

VI. RECENT AND FUTURE DEVELOPMENTS

Recently, we have implemented Version 2.0 of PANACEA in which several additional features have been incorporated.

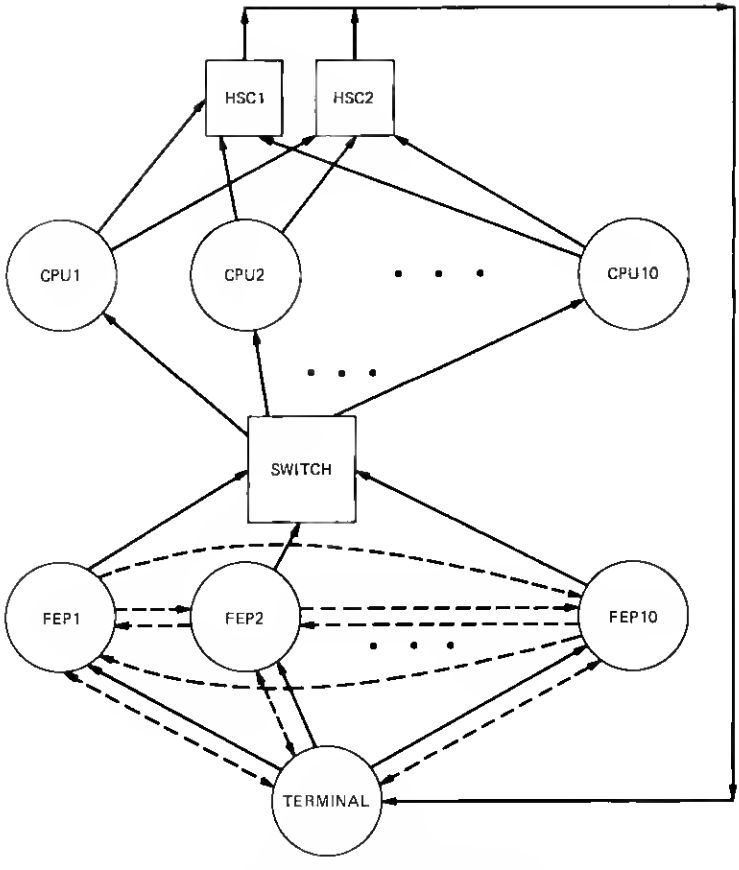


Fig. 9—Large network of Example 5.

(*i*) Version 2.0 solves mixed queueing networks in which some job classes, or possibly no job classes, are closed, while the remaining classes are open.

(*ii*) The computational method described in this paper is supplemented by other established recursive techniques.[13–17] The package selects the latter for small networks and the method of asymptotic expansions for all other networks.

Future work will be directed to:

(*i*) Heavy usage asymptotics
(*ii*) Incorporation of load-dependent servers
(*iii*) Sparsity exploitation in solving pseudo-networks
(*iv*) Priority queue approximations.

COMPUTATION PHASE BEGINS.

PACKAGE USES 1 TERMS IN THE ASYMPTOTICS FOR GOOD RESPONSE TIME ON THE VAX 11/780 HARDWARE

UTILIZATION STATISTICS ON PROCESSING NODES

UTILIZATION STATISTICS FOR NODE hsc2

CLASS	UTILIZATION	UPPER BOUND	LOWER BOUND
10	6.58941204e−02	6.60000106e−02	6.58941204e−02
11	6.59778304e−03	6.60000119e−03	6.59778304e−03
12	1.09938399e−02	1.10000014e−02	1.09938399e−02
13	5.99891352e−03	6.00000113e−03	5.99891352e−03
14	6.46839982e−01	6.60000114e−01	6.46839982e−01
15	6.59868518e−03	6.60000119e−03	6.59868518e−03
16	1.09963458e−02	1.10000014e−02	1.09963458e−02
17	5.99891352e−03	6.00000113e−03	5.99891352e−03

MEAN OUEUE LENGTH STATISTICS

STATISTICS FOR NODE hsc2

CLASS	MEAN OUEUE LENGTH	UPPER BOUND	LOWER BOUND
10	3.87797861e−01	3.87797861e−01	—
11	3.87797869e−02	3.87797869e−02	—
12	6.46329747e−02	6.46329747e−02	—
13	3.52543520e−02	3.52543520e−02	—
14	3.87797866e+00	3.87797866e+00	—
15	3.87797869e−02	3.87797869e−02	—
16	6.46329747e−02	6.46329747e−02	—
17	3.52543520e−02	3.52543520e−02	—

STANDARD DEVIATION STATISTICS ON OUEUE LENGTH

STATISTICS FOR NODE hsc2

CLASS	STD. DEVIATION	UPPER BOUND	LOWER BOUND
10	6.88753131e−01	7.90422721e−01	—
11	1.99112072e−01	2.02853370e−01	—
12	2.58917088e−01	2.66862288e−01	—
13	1.89657360e−01	1.92906147e−01	—
14	3.54066101e+00	5.25119023e+00	—
15	1.99112072e−01	2.02853370e−01	—
16	2.58917088e−01	2.66862288e−01	—
17	1.89657360e−01	1.92906147e−01	—

END OF OUTPUT STATISTICS.

Fig. 10—Output of Example 5.

REFERENCES

1. F. P. Kelly, *Reversibility and Stochastic Networks*, New York: John Wiley, 1980, Chapter 3.
2. L. Kleinrock, *Queueing Systems. Vol II: Computer Applications*, New York: John Wiley, 1976, Chapter 4.

3. M. Schwartz, *Computer-Communication Network Design and Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1977, Chapter 11.
4. F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers," J. of the ACM, *22*, No. 2 (April 1975), pp. 248–60.
5. F. R. Moore, "Computational Model of a Closed Queueing Network with Exponential Servers," IBM Journal of Research and Development, *16*, No. 6 (November 1972), pp. 567–72.
6. J. Buzen, "Queueing Network Models of Multiprogramming," Ph.D. Thesis, Div. of Eng. and App. Sci., Harvard University, Cambridge, MA: 1971.
7. R. R. Muntz and J. Wong, "Asymptotic Properties of Closed Queueing Network Models," Proc. 8th Annual Princeton Conf. on Info. Sci. and Systems (March 1974), pp. 348–52.
8. D. P. Gaver and G. S. Shedler, "Approximate Models for Processor Utilization in Multiprogrammed Computer Systems," SIAM J. on Computing, *2* (September 1973), pp. 183–92.
9. M. Reiser, "A Queueing Network Analysis of Computer Communication Networks with Flow Control," IEEE Trans. Commun., *COM-27*, No. 8 (August 1979), pp. 1199–1209.
10. S. S. Lam and M. Reiser, "Congestion Control of Store-and-Forward Networks by Input Buffer Limits-An Analysis," IEEE Trans. Commun., *COM-27*, No. 1 (January 1979), pp. 127–34.
11. B. Pittel, "Closed Exponential Networks of Queues with Saturation: The Jackson-Type Stationary Distribution and Its Asymptotic Analysis," Math. of Oper. Res., *4*, No. 4 (November 1979), pp. 357–78.
12. E. Arthurs and B. W. Stuck, "A Theoretical Performance Analysis of a Markovian Switching Node," IEEE Trans. Commun., *COM-26*, No. 11 (November 1978), pp. 1779–84.
13. J. P. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers," Commun. of tbe ACM, *16* (September 1973), pp. 527–31.
14. M. Reiser and H. Kobayashi, "Queueing Networks witb Multiple Closed Chains: Theory and Computational Algorithms," IBM J. of Res. and Dev., *19*, No. 3 (May 1975), pp. 283–94.
15. M. Reiser and S. S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," J. Assoc. Comput. Mach., *27* (April 1980), pp. 312–22.
16. C. H. Sauer and K. M. Chandy, "Computer Systems Performance Modeling, A Primer," Englewood Cliffs, NJ: Prentice Hall, 1981.
17. S. C. Bruell and G. Balbo, "Computational Algorithms for Closed Queueing Networks," New York: North-Holland, 1980.
18. J. McKenna, D. Mitra, and K. G. Ramakrishnan, "A Class of Closed Markovian Queueing Networks: Integral Representations, Asymptotic Expansions, Generalizations," B.S.T.J., *60*, No. 5 (May–June 1981), pp. 599–641.
19. J. McKenna and D. Mitra, "Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage," B.S.T.J., *61*, No. 5 (May–June 1982), pp. 661–83.
20. J. McKenna and D. Mitra, "Asymptotic Expansions and Integral Representation of Moments of Queue Lengths in Closed Markovian Networks," to be presented at the Int. Seminar on Modelling and Performance Evaluation Methodology, Paris, January 1983.
21. M. E. Lesk and E. Schmidt, unpublished work.
22. K. G. Ramakrishnan and D. Mitra, unpublished work.
23. G. M. Weinberg, *Psychology of Computer Programming*, New York: Van Nostrand, 1971.
24. Stephen C. Johnson, unpublished work.
25. "User's Manual for CADS," Austin, TX: Information Research Associates.